Interrupt Processing

Profs. Karkal Prabhu and Naga Kandasamy
ECE Department, Drexel University

This assignment consists of three problems, worth thirty points, dealing with interrupt processing. Complete the first problem in the lab and show the program's output to the teaching assistant to be checked for correctness. You may complete the remaining problems outside of lab and submit via BBLearn. You will find detailed submission instructions towards the end of this document.

Your C code must be clearly written, properly formatted, and well commented for full credit. Please submit original work.

Interrupt-Driven General Purpose IO (GPIO)

(10 points) Initialize a counter variable to 0. Each time BUTTON1 is pressed the counter should increment by 1, and each time BUTTON2 is pressed the counter should decrement by 1. If the counter value becomes 0, the RGB LED on the board, LED2 should be turned off. If the counter value is positive LED2 should light up green, and if the counter value is negative, LED2 should light up red.

Use a suitable interrupt service routine (ISR) to update the counter value based on button presses. Remember that both buttons are on PORT2 and therefore the same ISR should be used for both. You must examine the status flag to determine which button was pressed. The main processing loop must be responsible for setting the state of LED2. You may choose to print the value of counter to the terminal via the UART, also in the main processing loop.

Signature of the teaching assistant: ______________ Date: ______________

Finite State Machine

(10 points) Implement a finite state machine with eight states to light up the RGB LED on the MSP432 board, LED2, with different colors as follows.

State	LED2 color
0	Off
1	Red
2	Green
3	Blue
4	Yellow (Red + Green)
5	Purple (Red + Blue)
6	Cyan (Green + Blue)
7	White (Red + Green + Blue)

State transitions should be controlled by button presses as follows:

- BUTTON1: $0 \rightarrow 1, 2 \rightarrow 3, 4 \rightarrow 5$, and $6 \rightarrow 7$.
- BUTTON2: $1 \rightarrow 2, 3 \rightarrow 4, 5 \rightarrow 6$, and $7 \rightarrow 0$.

In other words, by pressing BUTTON1 and BUTTON2 alternately, LED2 should transition through the complete 8-color sequence. The common interrupt handler for BUTTON1 and BUTTON2 should handle the state transitions whereas the main processing loop will perform the initialization and then light up LED2 with the appropriate color depending on the state.

Handling UART and Button Inputs

(10 points) Write a program that lights up LED2 depending on a one-character color code that is provided by the user via the UART. The first letter of each color will serve as the color code; for example, 'O' for OFF, 'R' for red, 'G' for green, and so on. Your program must satisfy the following specifications:

- The system should begin accepting UART input only after the user first presses BUTTON1.
- The system should switch LED2 off and stop accepting UART input once BUTTON2 is pressed.

You will need two interrupt handlers: one for UART and another common one for BUTTON1 and BUTTON2. Enable the UART interrupts inside the interrupt handler for the buttons. Disable the UART interrupts after switching OFF LED2 also inside the button interrupt handler.

Submission Instructions

Once you have implemented all of the required features described for Problems 2 and 3, submit your code by doing the following:

- Run `Clean Project` to remove the executable and object files from the project folders. We must be able to build your projects from source and we don't require your pre-compiled executables or intermediate object files. **If your code does not at the very least compile, you will receive a zero.**
- Zip up each project separately and upload both zip files using the Blackboard Learn submission link found on the course website.